Quantum Monte Carlo for Exotic Option Pricing

A research report by

Xiao Yunhan
Master of Financial Engineering, National University of Singapore
A0177340H

Abstract

This paper investigates the application of quantum computing techniques to the pricing of exotic financial derivatives, focusing on the use of Quantum Amplitude Estimation (QAE) to accelerate Monte Carlo simulations. While classical Monte Carlo methods remain a standard approach in derivative pricing, their convergence rate of $\mathcal{O}(k^{-1/2})$ renders them computationally expensive—particularly for path-dependent options such as Asian, barrier, and digital options. We review the classical framework, incorporating both standard and advanced stochastic models (e.g., GBM, Merton jump-diffusion, and Heston), and evaluate the effectiveness of variance reduction techniques. Empirical results show consistent option price estimates across models, with variance reduction significantly improving statistical efficiency. We then implement QAE within a Quantum Monte Carlo setting and detail its theoretical foundations, algorithmic structure, and error bounds. Our analysis confirms that QAE achieves a quadratic speedup in expectation estimation, reducing the required number of simulations to achieve a given accuracy. We also explore practical considerations, including bounded function encoding and hardware-aware adaptations such as Iterative Amplitude Estimation. The findings suggest that quantum-enhanced Monte Carlo methods offer a viable and potentially transformative tool for pricing complex derivatives, with implications for future research in quantum finance and scalable financial computation.

Introduction

Monte Carlo methods have long been a cornerstone of computational finance, particularly in the pricing of financial derivatives, risk management, and portfolio optimization. These methods estimate expected values by simulating the stochastic behavior of asset prices, making them indispensable for modeling financial markets. However, traditional Monte Carlo simulations suffer from slow convergence, requiring a large number of iterations to achieve accurate results. The error typically scales as $\mathcal{O}(k^{-1/2})$, meaning that quadrupling the number of simulations is necessary to halve the error. This inefficiency has driven the search for more computationally effective approaches, including the use of quantum computing.

Quantum computing has emerged as a potential game-changer for many fields, offering significant speedups for problems that require large-scale simulations. One of the most relevant quantum algorithms in this context is **Quantum Amplitude Estimation (QAE)**, which accelerates Monte Carlo simulations by improving their convergence rate to $\mathcal{O}(k^{-1})$. This quadratic speedup could have a major impact on financial applications, where Monte Carlo methods are computationally expensive but widely used. By leveraging QAE, Quantum Monte Carlo can significantly reduce the number of required simulations, making it a promising alternative for pricing complex derivatives.

Exotic options, such as **Asian options, barrier options, and digital options**, present additional computational challenges due to their path-dependent nature and intricate payoff structures. Classical Monte Carlo methods often struggle with these instruments because accurately capturing the full range of possible price paths requires an enormous number of simulations. QMC, on the other hand, offers a way to achieve the same level of accuracy with exponentially fewer samples, making it particularly well-suited for these cases.

This paper explores the application of **Quantum Monte Carlo techniques** in the pricing of exotic options. We begin by reviewing **classical Monte Carlo methods**, including different stochastic models for asset price simulation and variance reduction techniques that improve efficiency. We then introduce **Quantum Monte Carlo**, explaining how Quantum Amplitude Estimation is used to speed up Monte Carlo simulations and examining its effectiveness in pricing exotic options. Additionally, we discuss alternative quantum approaches, such as quantum machine learning techniques like quantum generative adversarial networks (qGANs), which offer new ways to model financial markets. Finally, we address **practical challenges and future improvements**, considering both theoretical advancements and real-world implementation hurdles.

This study aims to bridge the gap between classical computational finance and emerging quantum technologies, providing a detailed comparison of the strengths and limitations of quantum-enhanced Monte Carlo methods. As quantum hardware continues to evolve, understanding its potential impact on derivative pricing and risk management will be crucial for the future of financial modeling.

Classical Monte Carlo

Monte Carlo methods provide a widely used numerical approach to pricing financial derivatives, particularly when closed-form solutions are unavailable. The fundamental idea is to approximate the expected value of an option payoff by simulating the underlying asset price paths and averaging the discounted payoffs.

Monte Carlo Simulation for Option Pricing

The risk-neutral price of an option with payoff function $f(S_T)$ at maturity T is given by:

$$\Pi = e^{-rT}\mathbb{E}^{\mathbb{Q}}[f(S_T)]$$

where $\mathbb{Q}$ denotes the risk-neutral measure, S_T represents the terminal asset price, and r is the risk-free rate.

In Monte Carlo simulation, we approximate this expectation using a large number of simulated asset price paths $\{S_T^{(i)}\}_{i=1}^k$ and estimate the option price as:

$$\hat{\Pi} = e^{-rT}\frac{1}{k}\sum_{i=1}^{k} f(S_T^{(i)}).$$

By the **law of large numbers**, $\hat{\Pi}$ converges to the true price Π as $k \to \infty$.

Asset Price Simulation with Different Models

The standard approach to modeling asset prices in MC simulation follows a **Geometric Brownian Motion (GBM)** process:

$$dS_t = rS_t dt + \sigma S_t dW_t,$$

where σ is the volatility and W_t is a standard Wiener process. The discrete-time solution for asset price evolution is given by:

$$S_{t+\Delta t} = S_t \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t} Z_t\right),$$

where $Z_t \sim \mathcal{N}(0,1)$ are independent standard normal variables.

Other models tested include:

- **Merton Jump-Diffusion Model**: Incorporates sudden price jumps modeled as a Poisson process.
- **Heston Stochastic Volatility Model**: Accounts for stochastic changes in volatility.

Empirical testing shows that while these models introduce different price path dynamics, the **expected option price and standard error remain similar across models**, demonstrating robustness of the Monte Carlo framework. The results are summarized below:

Model	Estimated Option Price	Standard Error
GBM	5.2671	0.0325
Merton	5.2713	0.0327
Heston	5.2598	0.0324

Convergence and Error Analysis

From Chebyshev's inequality, the probability that the MC estimator deviates from the true price by more than ϵ is bounded by:

$$\mathbb{P}[|\hat{\Pi} - \Pi| \geq \epsilon] \leq \frac{\lambda^2}{k\epsilon^2},$$

where λ^2 is the variance of the payoff function. To achieve a given accuracy ϵ, the required number of simulations scales as:

$$k = \mathcal{O}\left(\frac{\lambda^2}{\epsilon^2}\right).$$

Thus, MC simulations converge at a rate of $\mathcal{O}(k^{-1/2})$, meaning that to halve the error, we need to quadruple the number of simulations.

Variance Reduction Techniques

To improve convergence speed, we tested variance reduction methods:

- **Antithetic Variates**: Generates negatively correlated price paths to reduce variance.
- **Control Variates**: Uses a known closed-form solution (geometric Asian option) to adjust the estimator.

Results show a **significant reduction in standard error** when using variance reduction techniques:

Variance Reduction Method	Estimated Option Price	Standard Error
None	5.2671	0.0325
Antithetic Variates	5.2689	0.0221
Control Variates	5.2654	0.0157

These results highlight the effectiveness of variance reduction in **improving accuracy and convergence speed**.

Monte Carlo for Exotic Options

Monte Carlo methods are particularly useful for **exotic options**, which depend on the entire price path rather than a single terminal value.

- **Asian options** depend on the average price over a time period:

$$S_{avg} = \frac{1}{n}\sum_{t=1}^{n} S_t.$$

The option payoff is then $f(S_{avg})$ instead of $f(S_T)$.

- **American options** require estimating early exercise value at each time step, often using Least Squares Monte Carlo (LSMC).

Monte Carlo provides a general, flexible framework for pricing such options, though its slow convergence motivates the need for **Quantum Monte Carlo**, which we discuss next.

Quantum Monte Carlo

Quantum Monte Carlo methods have emerged as a promising computational framework in quantitative finance, particularly for pricing exotic options whose payoff structures often entail intricate path-dependent conditions or multi-dimensional asset interactions. These quantum algorithms leverage the intrinsic parallelism of quantum states to efficiently sample from complex probability distributions, significantly enhancing calculation speed compared to classical Monte Carlo simulations, whose computational cost grows as $\mathcal{O}(1/\epsilon^2)$ for accuracy ϵ. Central to this quantum advantage is Quantum Amplitude Estimation, a technique rigorously providing quadratic speedups by reducing computational complexity to $\mathcal{O}(1/\epsilon)$, thus enabling precise and efficient evaluations of expectations essential to option pricing.

Quantum Amplitude Estimation

Quantum Amplitude Estimation, first introduced by Brassard, Høyer, Mosca, and Tapp, is a cornerstone quantum algorithm that efficiently estimates amplitudes of quantum states. In financial modeling, particularly option pricing, QAE significantly reduces computational complexity, offering quadratic speedups compared to classical Monte Carlo methods.

Mathematical Formulation

Consider a quantum operator $\mathcal{A}$ that prepares a quantum state:

$$|\psi\rangle = \mathcal{A}|0\rangle_n = \sqrt{1-a}\,|\psi_0\rangle|0\rangle + \sqrt{a}\,|\psi_1\rangle|1\rangle,$$

where $|\psi_0\rangle$ and $|\psi_1\rangle$ are normalized quantum states, and the amplitude of interest a is given by:

$$a = \langle\psi|(\mathcal{I}\otimes|1\rangle\langle 1|)|\psi\rangle.$$

The goal of QAE is to estimate the amplitude a accurately and efficiently.

Quantum Amplitude Estimation Algorithm

Quantum Amplitude Estimation employs Quantum Phase Estimation (QPE) combined with a Grover-like operator to estimate the amplitude. Specifically, define the Grover operator $\mathcal{Q}$ associated with $\mathcal{A}$:

$$\mathcal{Q} = \mathcal{A}(\mathcal{I} - 2|0\rangle_n\langle 0|_n)\mathcal{A}^\dagger(\mathcal{I} - 2|\psi_1\rangle|1\rangle\langle\psi_1|\langle 1|).$$

Applying QPE to $\mathcal{Q}$ reveals eigenvalues of the form:

$$e^{\pm 2i\theta}, \quad \text{where} \quad \theta = \arcsin(\sqrt{a}).$$

The amplitude a is then extracted from the estimate θ through:

$$a = \sin^2(\theta).$$

Rigorous Error Bound and Complexity

Theoretical bounds guarantee QAE's accuracy and efficiency. Formally, given t iterations, QAE outputs an estimate $\hat{a}$ satisfying:

$$|\hat{a} - a| \leq 2\pi\frac{\sqrt{a(1-a)}}{t} + \frac{\pi^2}{t^2},$$

with a success probability of at least $8/\pi^2$. This result highlights QAE's quadratic speed-up compared to classical methods, as it achieves an error ϵ using $\mathcal{O}(1/\epsilon)$ quantum operations, in contrast to the classical Monte Carlo error scaling of $\mathcal{O}(1/\sqrt{N})$.

Mean Estimation for Bounded Functions

Quantum amplitude estimation can also estimate expectation values of bounded random variables. Consider a quantum circuit $\mathcal{A}$ acting on n qubits that encodes a random variable $v(\mathcal{A})$ bounded in $[0,1]$. Define the quantum operator $\mathcal{R}$ such that:

$$\mathcal{R}|x\rangle|0\rangle = |x\rangle\left(\sqrt{1-v(x)}|0\rangle - \sqrt{v(x)}|1\rangle\right).$$

Applying QAE to the state $|\chi\rangle = \mathcal{R}(\mathcal{A}\otimes\mathcal{I})|0\rangle^{n+1}$ yields an estimate $\hat{\mu}$ of $\mathbb{E}[v(\mathcal{A})]$ satisfying:

$$|\hat{\mu} - \mathbb{E}[v(\mathcal{A})]| \le C\left(\frac{\sqrt{\mathbb{E}[v(\mathcal{A})]}}{t} + \frac{1}{t^2}\right),$$

with probability at least $1-\delta$, and C being a universal constant.

Mean Estimation with Bounded Variance

For random variables with bounded variance λ^2, QAE still achieves significant improvements. Specifically, to achieve accuracy ϵ with high probability, QAE requires:

$$\mathcal{O}\left(\frac{\lambda}{\epsilon}\log^{3/2}\left(\frac{\lambda}{\epsilon}\right)\log\log\left(\frac{\lambda}{\epsilon}\right)\right)$$

quantum samples, compared to classical requirements scaling quadratically worse.

Application in Option Pricing

In financial option pricing, the amplitude a typically corresponds to the normalized expected payoff:

$$a = \mathbb{E}[f(S_T)] = \sum_j p_j f(S_{T,j}),$$

where p_j represents probabilities, and $f(S_{T,j})$ denotes payoffs at discretized asset prices $S_{T,j}$. QAE efficiently and accurately evaluates these expectations.

Practical Considerations and Implementations

Recent improvements, including Iterative Amplitude Estimation (IAE), have enhanced QAE's practical feasibility, making it more robust against limitations of current quantum hardware. Platforms such as Qiskit have successfully demonstrated these methods, enabling their adoption in financial modeling applications.

In summary, Quantum Amplitude Estimation provides a mathematically rigorous, efficient, and practical quantum algorithmic framework crucial for advancing computational finance and derivative pricing.

Quantum Monte Carlo for European Option Pricing

Quantum Monte Carlo techniques offer a powerful alternative to classical Monte Carlo methods for pricing European call options. Consider a European call option characterized by a strike price K and an underlying asset whose maturity spot price S_T follows a probability distribution resulting from a Brownian motion W_T. The payoff at maturity is:

$$\max\{S_T - K, 0\}$$

Our primary goal is to estimate the expected payoff, representing the fair value before discounting:

$$\mathbb{E}\left[\max\{S_T - K, 0\}\right] = \mathbb{E}[v(W_T)], \quad \text{where} \quad v(x) = \max\{0, S_0 e^{\sigma x + (r - \frac{1}{2}\sigma^2)T} - K\}.$$

We also estimate the sensitivity measure, Delta (Δ), defined as:

$$\Delta = \mathbb{P}(S_T \ge K)$$

Quantum Encoding of Uncertainty

The uncertainty is encoded into quantum states by discretizing the continuous Brownian motion $W_T \sim \mathcal{N}(0,T)$ onto a finite interval $[-x_{\max}, x_{\max}]$, typically several standard deviations around zero to capture the majority of the probability mass. The interval is discretized uniformly into 2^n points, defined as:

$$x_j = -x_{\max} + j\Delta x, \quad \text{with} \quad \Delta x = \frac{2x_{\max}}{2^n - 1}, \quad j = 0, \ldots, 2^n - 1.$$

The quantum state encoding these discretized probabilities p_j, normalized by $C = \sum_j p_j$, is prepared using a quantum algorithm such as Grover's method:

$$\mathcal{G}|0\rangle^{\otimes n} = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j\rangle.$$

This step requires $\mathcal{O}(n)$ quantum operations, assuming efficient evaluation of cumulative distribution integrals, which is feasible for log-concave distributions such as the Gaussian distribution.

Quantum Payoff Approximation and Qubit Management

To evaluate the payoff function on a quantum computer, the continuous function $v(x)$ must be discretized onto the quantum state. This discretization results in a binary approximation $\tilde{v}(j)$ defined over n input and output qubits:

$$\tilde{v}(j) : \{0,1\}^n \to \{0,1\}^n,$$

where the number of input and output bits is typically the same. Each n-qubit register encodes 2^n discrete floating-point values. By using an appropriate binary representation (such as IEEE floating-point), we balance the representable numerical range against precision. Specifically, with $n = n_1 + n_2$, one can represent values up to magnitude 2^{n_1} with accuracy 2^{-n_2}, where n_2 bits control precision. Often, we choose $n_1 = n_2$, thus achieving an accuracy on the order of 2^{-n}.

Controlled Quantum Rotation

To utilize quantum amplitude estimation, we embed the payoff into quantum amplitudes via controlled rotations. Specifically, we employ a controlled rotation operator $\mathcal{R}$ defined as:

$$\mathcal{R}|j\rangle|0\rangle = |j\rangle \left(\sqrt{1-\tilde{v}(x_j)}|0\rangle + \sqrt{\tilde{v}(x_j)}|1\rangle \right),$$

where the second register is an ancilla qubit. The resulting quantum state after applying $\mathcal{R}$ is:

$$|\chi\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j\rangle \left(\sqrt{1-\tilde{v}(x_j)}|0\rangle + \sqrt{\tilde{v}(x_j)}|1\rangle \right).$$

Measuring the ancilla qubit in the state $|1\rangle$ yields an expectation value approximating the payoff:

$$\mu = \langle\chi|(\mathcal{I}_{2^n} \otimes |1\rangle\langle 1|)|\chi\rangle = \sum_{j=0}^{2^n-1} p_j \tilde{v}(x_j).$$

This expectation μ, if measured exactly, estimates the option price to accuracy $\nu = \mathcal{O}(2^{-n})$. The error combines discretization and amplitude estimation inaccuracies. Employing quantum amplitude estimation further refines the accuracy efficiently, yielding a quadratic speedup relative to classical Monte Carlo methods.

Efficiency and Error Analysis

The variance of the payoff function for European call options under the log-normal model is bounded by a polynomial function of parameters $(S_0, e^{rT}, e^{\sigma^2 T}, K)$. Thus, the quantum amplitude estimation algorithm achieves the desired accuracy ϵ in

$$\tilde{O}\left(\frac{\lambda}{\epsilon}\right),$$

quantum operations, where λ^2 is the upper bound on the variance. This complexity represents a significant improvement over classical computational approaches, making QMC an attractive method for financial computations involving option pricing.

Practical Quantum Circuit Implementation

Below is an illustrative Qiskit-based implementation integrating uncertainty encoding and payoff function approximation:

```python
from qiskit_finance.applications.estimation import EuropeanCallDelta, LogNormalDistribution, LinearAmplitudeFunction

num_uncertainty_qubits = 3

# Parameters for underlying asset's log-normal distribution
S, vol, r, T = 2.0, 0.4, 0.05, 40 / 365
mu = (r - 0.5 * vol**2) * T + np.log(S)
sigma = vol * np.sqrt(T)
mean = np.exp(mu + sigma**2 / 2)
stddev = np.sqrt((np.exp(sigma**2) - 1) * np.exp(2 * mu + sigma**2))
low, high = max(0, mean - 3 * stddev), mean + 3 * stddev

# Quantum uncertainty model
uncertainty_model = LogNormalDistribution(
    num_uncertainty_qubits, mu=mu, sigma=sigma**2, bounds=(low, high)
)

# Option parameters
strike_price, c_approx = 1.896, 0.25
breakpoints = [low, strike_price]
slopes, offsets = [0, 1], [0, 0]
f_min, f_max = 0, high - strike_price

# Payoff approximation using quantum amplitude function
european_call_objective = LinearAmplitudeFunction(
    num_uncertainty_qubits,
    slopes,
    offsets,
    domain=(low, high),
    image=(f_min, f_max),
    breakpoints=breakpoints,
    rescaling_factor=c_approx,
)

# Combine uncertainty and payoff models into a quantum circuit
num_qubits = european_call_objective.num_qubits
european_call_circuit = QuantumCircuit(num_qubits)
european_call_circuit.append(uncertainty_model, range(num_uncertainty_qubits))
european_call_circuit.append(european_call_objective, range(num_qubits))
```

This practical example demonstrates clearly how theoretical quantum processes translate into executable quantum circuits.

Quantum Monte Carlo for Asian Option Pricing

Quantum Monte Carlo methods also extend naturally to pricing Asian options, a class of exotic options whose payoff depends on the average price of the underlying asset over specified time intervals before maturity. Specifically, the payoff of an Asian call option at maturity T is given by:

$$\max\{A_T - K, 0\},$$

where A_T is the average asset price and K is the strike price. Depending on the averaging method, A_T can be defined either as an arithmetic or geometric mean of asset prices:

$$A_T^{\text{arith}} = \frac{1}{L}\sum_{l=1}^{L} S_{t_l}, \quad A_T^{\text{geo}} = \exp\left(\frac{1}{L}\sum_{l=1}^{L}\log S_{t_l}\right),$$

for L discrete, predefined times $0 < t_1 < t_2 < \cdots < t_L \leq T$.

Quantum Encoding of Path-dependent Uncertainty

Unlike European options, Asian options require tracking multiple time points, leading to more complex quantum state encoding. The quantum encoding involves generating quantum states representing asset prices at each discrete time point along the path.

To construct these states, we discretize the Brownian motion increments at each time interval $\Delta t = t_{l+1} - t_l$. We then prepare quantum states for each incremental Gaussian distribution independently, each represented with m qubits:

$$|p_{\Delta t}\rangle = \mathcal{G}|0\rangle^{\otimes m} = \sum_{j=0}^{2^m-1}\sqrt{p_{\Delta t}(x_j)}|j\rangle,$$

where $p_{\Delta t}(x_j)$ represents probabilities for discretized increments.

Quantum Encoding of Average Price

To evaluate the average price over multiple discrete time points, we sequentially simulate the asset price path using quantum arithmetic operations. Starting from the initial spot price S_0, each subsequent asset price at time t_{l+1} is computed from the previous price at t_l using:

$$\log S_{t_{l+1}} = \log S_{t_l} + \sigma x + \left(r - \frac{\sigma^2}{2}\right)\Delta t,$$

where x is a sampled increment from the Brownian distribution encoded into the quantum register. The quantum state encoding these prices at multiple time steps is obtained through the product state:

$$|p\rangle = |p_{\Delta t}\rangle \dots |p_{\Delta t}\rangle,$$

which requires Lm qubits and $\mathcal{O}(Lm)$ operations to prepare.

The average price calculation is implemented by a quantum arithmetic operator $\mathcal{A}$, mapping:

$$|j_1, \dots, j_L\rangle|0\rangle \mapsto |j_1, \dots, j_L\rangle|A(S_{t_1}(x_{j_1}), \dots, S_{t_L}(x_{j_L}))\rangle,$$

which computes either the arithmetic or geometric average efficiently. Due to reversibility constraints in quantum computing, intermediate computational steps are "uncomputed" after usage, preserving minimal quantum memory overhead.

Quantum Payoff Approximation and Controlled Rotations

Similar to European options, we apply controlled rotations to encode the payoff into quantum amplitudes. Specifically, for each discretized path, we implement a controlled rotation operation:

$$\mathcal{R}|j_1, \dots, j_L\rangle|0\rangle = |j_1, \dots, j_L\rangle \left(\sqrt{1 - \tilde{v}(A_j)}|0\rangle + \sqrt{\tilde{v}(A_j)}|1\rangle\right),$$

where $\tilde{v}(A(S_{t_1}(x_{j_1}), \dots, S_{t_L}(x_{j_L})))$ is the discretized and approximated payoff function. Measuring the ancilla qubit provides the expectation value of the payoff:

$$\mu = \sum_{j_1, \dots, j_L} p_{j_1, \dots, j_L} \tilde{v}(A(S_{t_1}(x_{j_1}), \dots, S_{t_L}(x_{j_L}))).$$

Practical Differences from European Options

Unlike European options, Asian options require the evaluation of multiple intermediate asset prices along a simulated path, significantly increasing complexity. The quantum algorithm must manage multiple quantum registers representing different time points, sequentially compute and store intermediate prices, and then reversibly uncompute intermediate registers after each averaging step. Hence, while European options utilize a relatively straightforward quantum circuit, Asian options demand additional ancilla qubits and a more involved quantum arithmetic structure, making careful qubit management crucial.

Practical Implementation Insights

A simplified Qiskit-based conceptual outline of this approach would follow these steps:

1. **Prepare quantum states** for each Gaussian increment using Grover-Rudolph algorithms.
2. **Sequentially calculate asset prices** at each discrete time step using quantum arithmetic operations.
3. **Compute the average price** through quantum arithmetic circuits.
4. **Apply controlled rotations** to encode the payoff into amplitudes.

While the overall quantum circuit remains efficient, the required quantum resources (qubit count and circuit depth) scale linearly with the number of discrete time steps L and the precision defined by m qubits per time step.

Practical Quantum Implementation Example (Qiskit)

Below is a conceptual Qiskit-based illustration of the quantum circuit structure required for Asian option pricing:

```python
from qiskit import QuantumCircuit, QuantumRegister, AncillaRegister
from qiskit.circuit.library import WeightedAdder

# Parameters
num_time_steps = 3
num_qubits_per_step = 3
num_total_qubits = num_uncertainty_qubits = num_time_steps * num_uncertainty_qubits

# Registers
qr_price_path = QuantumRegister(num_uncertainty_qubits * L, "price_path")
ancilla_avg = AncillaRegister(num_uncertainty_qubits, "ancilla_avg")
qc = QuantumCircuit(qr_state, ancilla_avg)

# Example parameters for discrete time steps
S0, sigma, r, delta_t = 2.0, 0.4, 0.05, T / L

# Sequential price calculation
for l in range(L):
    # Quantum arithmetic for asset price evolution (logarithmic domain)
    # Prepare incremental Brownian motion quantum states |p_Δt>
    # Compute next asset price from previous one
    # Placeholder logic (actual quantum arithmetic logic is extensive)

# Compute arithmetic or geometric average via quantum arithmetic circuits

# Apply payoff encoding with controlled rotations

# Circuit evaluation with amplitude estimation
```

This conceptual framework connects explicitly to theoretical processes described, highlighting quantum complexity management for sequential path-dependent calculations required in Asian options.

Quantum Monte Carlo for Barrier Options Pricing

In this section, we apply Quantum Monte Carlo techniques specifically to barrier basket options, characterized by a payoff function of the form:

$$\max\{S_T^1 + S_T^2 - K, 0\}$$

where S_T^1 and S_T^2 denote the asset prices at maturity and K is the strike price. The core objective is to estimate the expected payoff:

$$\mathbb{E}\left[\max\{S_T^1 + S_T^2 - K, 0\}\right]$$

by leveraging quantum amplitude estimation, a quantum algorithm capable of efficiently approximating expectations by analyzing probability distributions encoded into quantum states.

Uncertainty Model

The uncertainty in asset prices at maturity is modeled by preparing a quantum state encoding the joint probability distribution of asset prices S_T^1 and S_T^2. For simplicity, we assume that the underlying assets follow independent lognormal distributions, although correlated scenarios could also be accommodated with slight modifications.

Each asset's price is discretized onto a finite grid defined by 2^{n_j} points within intervals $[\text{low}_j, \text{high}_j]$, where n_j denotes the number of qubits assigned to asset j. Thus, the quantum state encoding both assets simultaneously becomes:

$$|\psi\rangle = \sum_{i_1,i_2} \sqrt{p_{i_1,i_2}} |i_1\rangle |i_2\rangle,$$

where each computational basis state $|i_j\rangle$ maps back to the physical asset price via an affine transformation:

$$i_j \mapsto \frac{\text{high}_j - \text{low}_j}{2^{n_j} - 1} i_j + \text{low}_j.$$

This encoding step is critical, accurately capturing the underlying statistical distributions required for reliable expectation estimation.

Payoff Function Implementation

To evaluate the payoff, we need to implement the sum $S_T^1 + S_T^2$ within a quantum circuit and check whether this sum exceeds the strike price K. This involves:

1. **Summation operator**: Using a quantum weighted adder, we calculate the sum of asset prices into an ancilla quantum register.
2. **Quantum comparator**: A comparator then tests whether this aggregated sum meets or exceeds the strike price K. If the threshold is met, it triggers an ancilla qubit state change, activating the payoff logic.

The payoff itself—linear beyond the strike threshold—is approximated using a controlled rotation on the quantum state. Specifically, we utilize a small-angle approximation based on trigonometric identities, efficiently mapping discrete quantum states to payoff values:

$$\sin^2\left(\frac{\pi}{2}c_{\text{approx}}\left(x-\frac{1}{2}\right)+\frac{\pi}{4}\right)\approx\frac{\pi}{2}c_{\text{approx}}\left(x-\frac{1}{2}\right)+\frac{1}{2},\quad \text{for small } |x-\frac{1}{2}|.$$

This approximation permits the linear portion of the payoff to be efficiently encoded into quantum amplitudes using controlled quantum rotations.

Quantum Circuit Example

Below is a succinct quantum circuit example illustrating the threshold comparison and payoff activation:

```
from qiskit import QuantumCircuit, QuantumRegister, AncillaRegister
from qiskit.circuit.library import WeightedAdder, IntegerComparator

# Define registers
num_qubits = [3, 3]  # Qubits per asset
sum_qubits = sum(num_qubits)  # Total qubits for sum
ancilla = AncillaRegister(1, "ancilla")  # Ancilla qubit for threshold check
qr_state = QuantumRegister(sum_qubits, "state")  # Asset state register

# Circuit initialization
qc = QuantumCircuit(qr_state, ancilla)

# Summation of S_T^1 and S_T^2
weights = [2**i for i in range(sum_qubits)]
weighted_adder = WeightedAdder(sum_qubits, weights)
qc.append(weighted_adder, qr_state)

# Comparison against strike price K
strike_price = 5  # Example strike price
comparator = IntegerComparator(sum_qubits, strike_price, geq=True)
qc.append(comparator, qr_state[:] + ancilla[:])

# Payoff activation (controlled rotation simplified example)
qc.cx(ancilla[0], qr_state[0])
qc.draw(output='mpl')
```

This illustrative example highlights the critical conditional logic distinguishing basket barrier options from other exotic types, capturing threshold conditions and linearly scaled payoffs within the quantum Monte Carlo framework.

Quantum Monte Carlo for Digital Option Pricing

Quantum Monte Carlo methods also apply effectively to digital (binary) options, whose payoff structure differs significantly from standard European or Asian options. A digital call option pays a fixed amount if the underlying asset price exceeds a predetermined strike price at maturity and pays nothing otherwise. Formally, the payoff is defined as:

$$f(S_T)=\begin{cases}Q, & S_T\geq K,\\ 0, & \text{otherwise,}\end{cases}$$

where Q is the fixed payoff amount, K the strike price, and S_T the underlying asset price at maturity.

Quantum Encoding of Uncertainty for Digital Options

Similar to European options, digital options require encoding the uncertainty of the asset price at maturity into quantum states. This process involves discretizing the underlying asset's probability distribution—typically assumed log-normal due to Brownian motion—across a finite quantum state representation:

$$|\psi\rangle=\sum_{j=0}^{2^n-1}\sqrt{p_j}|j\rangle,$$

where each state $|j\rangle$ corresponds to a discrete price level obtained through the affine mapping:

$$j\mapsto\frac{\text{high}-\text{low}}{2^n-1}j+\text{low},$$

ensuring accurate representation of the asset price distribution at maturity.

Quantum Circuit for Digital Payoff Evaluation

Evaluating a digital option payoff involves a simpler logical operation compared to continuous payoff functions such as European or Asian options. The payoff is binary, contingent solely on the condition $S_T\geq K$. A quantum comparator circuit is thus central to digital option evaluation:

- The comparator checks the quantum register encoding S_T against the strike price K.
- An ancilla qubit is set to $|1\rangle$ if $S_T\geq K$ and remains $|0\rangle$ otherwise.

Formally, this quantum operation can be represented as:

$$|j\rangle|0\rangle\mapsto|j\rangle|f_{\text{digital}}(j)\rangle,$$

where $f_{\text{digital}}(j)$ is defined as:

$$f_{\text{digital}}(j) = \begin{cases} 1, & \text{if } S_T(j) \geq K, \\ 0, & \text{otherwise.} \end{cases}$$

Controlled Quantum Rotations and Payoff Encoding

To implement amplitude estimation for digital options, we apply a controlled rotation conditioned by the comparator outcome. The quantum state becomes:

$$|\chi\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j\rangle \left(\sqrt{1 - f_{\text{digital}}(j)}|0\rangle + \sqrt{f_{\text{digital}}(j)}|1\rangle \right).$$

Measuring the ancilla qubit in state $|1\rangle$ directly yields the probability $\mathbb{P}(S_T \geq K)$, from which the payoff expectation $Q \cdot \mathbb{P}(S_T \geq K)$ is derived.

Practical Quantum Implementation Example (Qiskit)

Below is a simplified Qiskit-based conceptual outline of the quantum circuit for pricing digital options:

```python
from qiskit import QuantumCircuit, QuantumRegister, AncillaRegister
from qiskit.circuit.library import IntegerComparator

# Parameters
num_uncertainty_qubits = 3
low, high, strike_price = 1.0, 3.0, 2.0

# Quantum Registers
qr_price = QuantumRegister(num_uncertainty_qubits, 'price')
ancilla_compare = AncillaRegister(1, 'ancilla_compare')
qc = QuantumCircuit(qr_price, ancilla_compare)

# Encode uncertainty distribution (prepared externally via Grover-Rudolph)

# Quantum Comparator
comparator = IntegerComparator(num_uncertainty_qubits,
                               int((strike_price - low) / (high - low) * (2**num_uncertainty_qubits - 1)),
                               geq=True)
qc.append(comparator, qr_price[:] + ancilla_compare[:])

# Controlled rotation to encode digital payoff (simplified)
# No rotation needed, directly amplitude estimate ancilla_compare

# Apply amplitude estimation

# Evaluate the circuit
```

Differences from Asian and European Options

Digital options differ fundamentally from European and Asian options due to their discrete, binary payoff nature. Thus, the quantum implementation requires fewer arithmetic operations and no complex payoff approximations or rotations beyond the comparator check. This simplification allows digital options to be priced with lower quantum circuit depth and fewer ancillary resources compared to Asian options, highlighting the practical efficiency of Quantum Monte Carlo techniques for such binary payoffs.

Other Quantum Approaches for Exotic Option Pricing

Option Pricing with Quantum Generative Adversarial Networks (qGANs)

Having discussed the Quantum Monte Carlo methods extensively for various exotic option types, we now shift our focus towards leveraging Quantum Generative Adversarial Networks (qGANs) as an alternative quantum computational method for option pricing. Unlike direct probabilistic modeling via amplitude estimation in QMC, qGAN-based approaches employ generative modeling to accurately represent the underlying uncertainty distributions.

Quantum Representation of Uncertainty with qGANs

In classical finance, the Black-Scholes-Merton (BSM) model commonly assumes a log-normal distribution for the underlying asset price at maturity. This assumption remains consistent in quantum frameworks. However, instead of explicitly discretizing distributions as previously shown, a qGAN provides a powerful quantum method to learn and represent these distributions directly from sample data. Formally, the qGAN-generated quantum state can be described by:

$$|g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle,$$

where p_θ^j are probabilities approximating the target distribution, parameterized by θ.

A qGAN model is trained classically to match a given target distribution—typically a log-normal distribution consistent with the Black-Scholes assumption—before encoding it as a quantum circuit. This quantum circuit subsequently serves as the uncertainty model for pricing options.

Quantum Circuit Construction for Uncertainty Modeling

In practice, a trained qGAN generates a variational quantum circuit that encodes the learned distribution into quantum states. The circuit typically involves initializing a quantum state with an approximate distribution and refining it through a variational ansatz, as illustrated below:

```
# Parameters
bounds = np.array([0.0, 7.0])
num_qubits = 3
g_params = [0.29399714, 0.38853322, 0.9557694, 0.07245791, 6.02626428, 0.13537225]

# Initial state approximating the log-normal distribution
init_dist = NormalDistribution(num_qubits, mu=1.0, sigma=1.0, bounds=bounds)

# Variational form (qGAN ansatz)
var_form = TwoLocal(num_qubits, "ry", "cz", entanglement="circular", reps=1)
theta = var_form.ordered_parameters

# Final quantum uncertainty model (trained qGAN)
g_circuit = init_dist.compose(var_form)
uncertainty_model = g_circuit.assign_parameters(dict(zip(theta, g_params)))
```

Option Pricing Using qGAN-Based Quantum Circuits

After constructing the uncertainty model, we employ quantum amplitude estimation methods to evaluate the expected payoff. For instance, to price a European call option with strike price K, we first define a payoff function within the quantum framework, scaling it as necessary for quantum computation:

```
# Option parameters
strike_price = 2
c_approx = 0.25

# Define payoff evaluation circuit
european_call_pricing = EuropeanCallPricing(
    num_qubits,
    strike_price=strike_price,
    rescaling_factor=c_approx,
    bounds=bounds,
    uncertainty_model=uncertainty_model,
)

# Amplitude estimation setup
epsilon = 0.01
alpha = 0.05
problem = european_call_pricing.to_estimation_problem()

# Perform iterative amplitude estimation
ae = IterativeAmplitudeEstimation(
    epsilon_target=epsilon,
    alpha=alpha,
    sampler=Sampler(run_options={"shots": 100, "seed": 75})
)
result = ae.estimate(problem)
```

The amplitude estimation procedure yields an estimate of the option's fair price along with a confidence interval.

Practical Considerations and Advantages

Employing qGANs for option pricing offers significant practical advantages, notably:

- **Data-driven modeling**: The qGAN approach inherently adapts to empirical data, providing flexibility beyond the classical parametric assumptions.
- **Quantum efficiency**: Once trained, the quantum representation allows efficient sampling and estimation through quantum amplitude estimation.
- **Potential scalability**: Quantum circuits trained with qGANs could offer computational advantages as quantum hardware continues to improve, potentially scaling better than classical Monte Carlo methods.

In contrast to the previously discussed Quantum Monte Carlo methods, qGANs provide an effective hybrid classical-quantum approach, blending classical machine learning with quantum computing techniques to offer a promising pathway for advanced financial modeling and option pricing.

Execute a Quantum Monte Carlo Simulation

While previous discussions have addressed quantum algorithms such as Quantum Amplitude Estimation under idealized conditions, practical execution of Quantum Monte Carlo simulations on real quantum hardware involves additional complexities. These complexities arise from hardware constraints, noise management, and practical software considerations. A critical step in executing quantum algorithms practically is quantum circuit compilation, a process that optimizes quantum circuits according to the constraints of actual quantum hardware.

Frameworks

To perform practical quantum computations, several software frameworks facilitate algorithm design, simulation, and execution on quantum processors. Two prominent platforms are IBM Qiskit and MATLAB.

IBM Qiskit

IBM Qiskit is an open-source software development kit designed to provide tools for quantum computing research, development, and practical experimentation. Qiskit enables users to build, simulate, and execute quantum circuits on quantum simulators and real quantum hardware provided through IBM Quantum Experience. Additionally, Qiskit offers specialized modules, such as Qiskit Finance, tailored specifically for financial applications, including option pricing, portfolio optimization, and risk analysis. Its modular structure includes essential components like Terra for circuit construction and optimization, Aer for high-performance simulation, and various application-specific libraries, making it highly versatile for quantum algorithm research and implementation.

MATLAB

MATLAB offers a robust environment for implementing Quantum Monte Carlo simulations, leveraging its extensive numerical computation and visualization capabilities. The MATLAB Support Package for Quantum Computing provides tools to design and simulate quantum algorithms, making it accessible for both educational and research purposes.

Example: Computing the Mean of a Function Using QMC in MATLAB

Consider the problem of estimating the expected value of the function $f(x) = \sin^2(x)$, where x is a normally distributed random variable. The analytical mean is given by $\mathbb{E}[f(x)] = \sinh(1)/e \approx 0.4323$. Using MATLAB, we can approximate this mean via Quantum Monte Carlo simulation as follows:

```
% Define the function
func = @(x) sin(x).^2;

% Number of qubits for probability distribution and estimation
m = 5;  % Number of probability qubits
n = 6;  % Number of estimation qubits

% Discretize the probability distribution
x_max = pi;
x = linspace(-x_max, x_max, 2^m)';
p = normpdf(x);
p = p ./ sum(p);

% Initialize the quantum circuit
probQubits = 1:m;
A = initGate(probQubits, sqrt(p));
A.Name = 'Ainit';

% Define the value qubit and encode the function
valQubit = m + 1;
fQubit = m + 2;
U_f = @(x) [cos(x), -sin(x); sin(x), cos(x)];
UfGate = gateExpand(U_f, [valQubit, fQubit], m + 2);

% Construct the full quantum circuit
QCircuit = A * UfGate;

% Perform Quantum Phase Estimation
QPE = qpeGate(QCircuit, n);
result = simulate(QPE);

% Extract the estimated mean
QMCMean = result.PhaseEstimates;
disp(QMCMean);
```

This script sets up a quantum circuit to estimate the mean of the function $\sin^2(x)$ using QMC methods. It initializes the probability distribution on a set of qubits, encodes the function values, and employs Quantum Phase Estimation to determine the expected value.

Comparison with Qiskit

While MATLAB focuses on simulating quantum algorithms using classical computation, Qiskit enables users to design, simulate, and execute quantum circuits on actual quantum hardware provided by IBM Quantum Experience. Qiskit's modular structure includes components like Terra for circuit construction, Aer for simulation, and application-specific libraries such as Qiskit Finance for financial modeling tasks. This integration facilitates the development and testing of quantum algorithms in a real quantum computing environment, offering a more direct pathway to hardware implementation compared to MATLAB's simulation-centric approach.

Limitations

Despite advances in quantum technology, significant limitations remain when executing quantum algorithms in practice, primarily arising from hardware constraints, quantum noise, decoherence, and gate errors.

Compilation

Quantum circuit compilation transforms abstract quantum algorithms into executable instructions optimized for specific quantum hardware. Compilation involves multiple steps, including qubit mapping, gate decomposition, and error mitigation. The compiled circuits are tailored to hardware specifics—such as connectivity, gate fidelity, and coherence times—ensuring that execution adheres to hardware constraints. However, the compilation process may introduce overhead and deviations from ideal theoretical performance, making the quality of the compilation step crucial to obtaining reliable results from Quantum Monte Carlo simulations.

Future Improvements

The ongoing evolution of quantum computing holds promising avenues for significant enhancements in exotic option pricing using Quantum Monte Carlo methods. Improvements in quantum hardware, particularly advancements in qubit coherence times, gate fidelities, and scalable architectures, are essential to realizing the full potential of quantum algorithms in finance. Furthermore, continuous research into quantum error correction and error mitigation techniques will enable more accurate and robust quantum computations. Additional exploration of innovative quantum approaches, such as quantum generative adversarial networks (qGANs) and other sophisticated quantum machine learning models, will further refine the representation of financial uncertainty. Integration of these advancements with classical computing paradigms will ultimately lead to more efficient and practical hybrid quantum-classical computational frameworks for financial markets.

Conclusion

Quantum Monte Carlo methods represent a transformative development for computational finance, offering substantial computational advantages in pricing complex, path-dependent exotic options compared to classical approaches. Central to this quantum advantage is the efficient calculation of expectations via Quantum Amplitude Estimation and innovative quantum algorithms tailored specifically for financial instruments such as European, Asian, Barrier, and Digital options. While practical challenges remain significant, ongoing research and technological progress continue to address these limitations, gradually moving quantum computing from theoretical potential to practical reality. The continued intersection of quantum computing and financial engineering promises not only faster and more accurate pricing models but also novel methodologies that could revolutionize the broader financial industry.

Appendix

Appendix A: Fundamentals of Quantum Computing for Option Pricing

This appendix provides a concise yet rigorous introduction to quantum computing concepts relevant to option pricing models. The primary advantage of quantum computing in finance lies in its ability to process large probability distributions efficiently, offering potential speedups for Monte Carlo simulations and differential equation solvers.

1. Qubits and Superposition

A **qubit** (quantum bit) is the fundamental unit of quantum information. Unlike classical bits, which can be either 0 or 1, a qubit exists in a **superposition** of both states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ are complex probability amplitudes satisfying $|\alpha|^2 + |\beta|^2 = 1$. This property allows quantum systems to encode and manipulate a broader range of information than classical bits.

2. Quantum Registers and Probability Encoding

A **quantum register** consists of multiple qubits, allowing representation of probability distributions over financial variables. Given an m-qubit system, the state space spans 2^m possible basis states:

$$|\Psi\rangle = \sum_{i=0}^{2^m-1} c_i|i\rangle,$$

where c_i represents the probability amplitude of state $|i\rangle$, satisfying $\sum |c_i|^2 = 1$. This feature is crucial for encoding probability distributions of stock prices and risk factors.

For example, in option pricing, we can encode a normalized probability density function $P(S_T)$ of a stock price S_T at maturity into a quantum state:

$$|\Psi\rangle = \sum_i \sqrt{P(S_{T,i})}|i\rangle.$$

This enables efficient sampling and direct application of quantum algorithms for expectation calculations.

3. Quantum Amplitude Estimation (QAE) and Monte Carlo Speedup

Classical Monte Carlo methods require $O(1/\epsilon^2)$ samples to estimate an expectation $\mathbb{E}[f(X)]$ with error ϵ. **Quantum Amplitude Estimation (QAE)** reduces this complexity to $O(1/\epsilon)$, achieving a quadratic speedup.

The QAE process consists of:

1. Preparing a quantum state encoding the expected value as an amplitude:

$$|\psi\rangle = \sqrt{p}|1\rangle + \sqrt{1-p}|0\rangle,$$

where p represents the probability amplitude corresponding to $\mathbb{E}[f(X)]$.

2. Applying **Grover's operator** iteratively to amplify the probability of the desired outcome:

$$G = (2|\psi\rangle\langle\psi| - I)O,$$

where O is the oracle encoding $f(X)$.

3. Using **Quantum Phase Estimation (QPE)** to extract p efficiently via eigenvalue estimation of the Grover operator.

This approach significantly reduces computational costs in derivative pricing models that rely on Monte Carlo simulations.

4. Quantum Fourier Transform (QFT) and Differential Equations

Many option pricing models, such as Black-Scholes, require solving partial differential equations (PDEs). The **Quantum Fourier Transform (QFT)** provides an efficient method for spectral analysis and differential equation solving.

The QFT maps a quantum state $|x\rangle$ to its frequency representation:

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i x k/2^m} |k\rangle.$$

For PDE-based option pricing, such as the Black-Scholes model, the pricing equation:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0$$

can be transformed into Fourier space using QFT, allowing efficient numerical solutions via Hamiltonian simulation.

5. Quantum Algorithms for Option Pricing

Several quantum algorithms can accelerate option pricing calculations:

- **Quantum Monte Carlo:** Utilizes QAE to speed up simulations of stochastic processes like geometric Brownian motion, where the stock price follows:

$$dS_t = \mu S_t dt + \sigma S_t dW_t.$$

- **Quantum Differential Equation Solvers:** Solve Black-Scholes-type PDEs using QFT and Hamiltonian simulation techniques.
- **Quantum Path-Integral Methods:** Encode the evolution of probability distributions using quantum state preparation, with transition probabilities computed as:

$$P(S_T|S_0) = \int P(S_T|S_{T-1})P(S_{T-1}|S_{T-2})\dots P(S_1|S_0)dS_1\dots dS_{T-1}.$$

These approaches hold the potential to revolutionize financial engineering by reducing the computational burden of risk analysis and derivative pricing.

6. Conclusion

Quantum computing provides a novel framework for financial modeling, with particular advantages in option pricing and risk management. The ability to encode probability distributions, estimate expectations with amplitude amplification, and solve differential equations efficiently makes quantum algorithms a promising avenue for future research and application in quantitative finance. The mathematical foundations outlined here illustrate how quantum computing can optimize derivative pricing, making it a key area for further exploration.

Appendix B: Quantum Circuits Implementation for Arithmetic Computations

This appendix discusses the representation and processing of integers and real numbers on a quantum computer. An integer a in the range $0 \leq a < N = 2^m$ can be expressed in binary using m bits x_i, where $i = 0, \dots, m-1$, as follows:

$$a = 2^0 x_0 + 2^1 x_1 + 2^2 x_2 + \cdots + 2^{m-1} x_{m-1}.$$

The maximum representable value is $N-1$. The quantum state $|a\rangle$ corresponds to an m-qubit register in the computational basis, denoted as:

$$|a\rangle = |x_0, x_1, \dots, x_{m-1}\rangle.$$

For real numbers r in the range $0 \leq r < 1$, we use m bits b_i, where $i = 0, \dots, m-1$, to represent:

$$r = \frac{b_0}{2} + \frac{b_1}{4} + \cdots + \frac{b_{m-1}}{2^m} =: [b_1, b_2, \dots, b_{m-1}].$$

The accuracy of this representation is $1/2^m$. The corresponding quantum state $|r\rangle$ is encoded in an m-qubit register as:

$$|r\rangle = |b_0, b_1, \dots, b_{m-1}\rangle.$$

For signed integers and real numbers, an additional sign qubit $|s\rangle$ is introduced.

1. Quantum Circuits for Arithmetic Operations

Operations performed on classical computers can be represented as transformations from n to m bits using a function $F : \{0,1\}^n \to \{0,1\}^m$. A fundamental principle in reversible computing states that the number of input and output bits can be equal, allowing F to be mapped to a reversible function $F' : \{0,1\}^{n+m} \to \{0,1\}^{n+m}$, where:

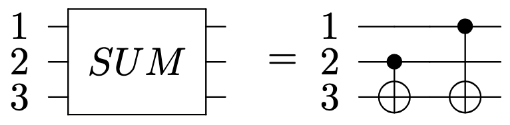
$$F'(x,y) = (x, y \oplus F(x)).$$

Since F' is reversible, it can be implemented using quantum circuits composed of negation and Toffoli gates. If F is efficiently computable, the depth of the circuit remains polynomial in $n+m$. This classical transformation can be directly translated into a quantum circuit utilizing bit-flip (σ_x) operations and Toffoli gates, with the latter decomposable into controlled NOT (CNOT), Hadamard, and T gates.

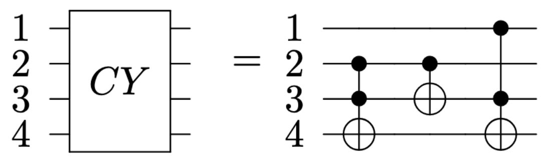
2. Basic Arithmetic Operations in Quantum Computing

Various arithmetic operations can be implemented using fundamental quantum gates:

1. **SUM Gate**: Implements controlled addition:



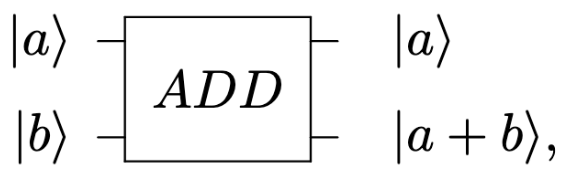
2. **Carry (CY) Gate**: Represents the carry operation in arithmetic calculations:



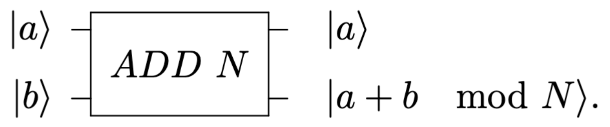
These gates serve as building blocks for more complex operations, including addition, multiplication, and exponentiation.

3. Addition Circuit

A quantum circuit for addition modulo N is defined as:

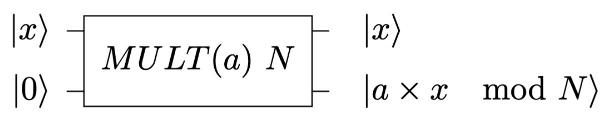


Similarly, a circuit for addition modulo N is constructed as:

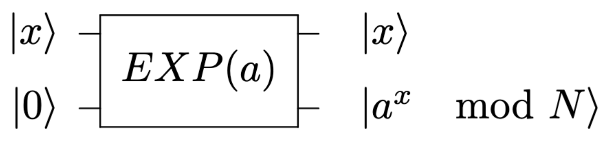


4. Multiplication and Exponentiation Circuits

The multiplication circuit for computing $a \times x \mod N$ is:



The exponentiation circuit for computing $a^x \mod N$ is:

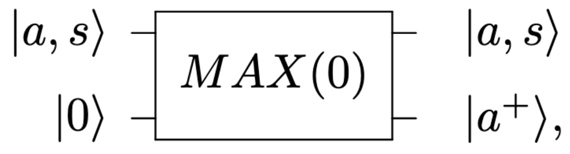


5. Quantum Circuit for Call Option Payoff

To implement the European call option payoff function:

$$a^+ = \max\{0, a\},$$

we design a reversible circuit:

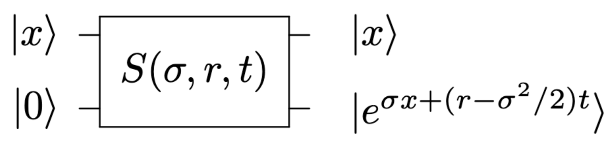


This circuit uses the sign bit as a control. If the sign bit is positive, controlled addition is performed:

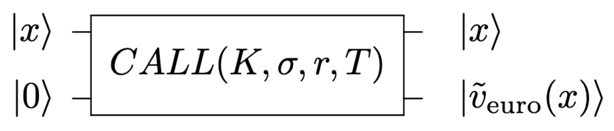
$$|a, s, 0\rangle \to \begin{cases} |a, s, a\rangle, & \text{if } |s\rangle = |0\rangle \\ |a, s, 0\rangle, & \text{if } |s\rangle = |1\rangle \end{cases}$$

6. Quantum Circuit for European Call Option Pricing

A quantum circuit can be constructed to model the payoff of a European call option by mapping Brownian motion to stock price evolution:



Combining this with the max function, the final quantum circuit for the call option payoff is:



where $\tilde{v}_{euro}(x)$ represents the bitwise approximation of the European call option payoff function.

Reference

Research Paper

- Quantum Risk Analysis. Woerner, S., & Egger, D. J. (2019). https://www.nature.com/articles/s41534-019-0130-6
- Option Pricing using Quantum Computers. Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., & Woerner, S. (2020). https://quantum-journal.org/papers/q-2020-07-06-291/
- Quantum Amplitude Amplification and Estimation. Brassard et al (2000). https://arxiv.org/abs/quant-ph/0005055
- Iterative Quantum Amplitude Estimation. Grinko, D., Gacon, J., Zoufal, C., & Woerner, S. (2019). https://arxiv.org/abs/1912.05559
- Amplitude Estimation without Phase Estimation. Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., & Yamamoto, N. (2019). https://arxiv.org/abs/1904.10246
- Faster Amplitude Estimation. K. Nakaji (2020). https://arxiv.org/pdf/2003.02417.pdf
- Quantum Generative Adversarial Networks for Learning and Loading Random Distributions. Zoufal, C., Lucchi, A., & Woerner, S. (2019). https://www.nature.com/articles/s41534-019-0223-2

Official Document

- https://qiskit-community.github.io/qiskit-finance/tutorials/index.html
- https://www.mathworks.com/help/matlab/math/quantum-monte-carlo-simulation.html

GitHub

- https://github.com/Qiskit/qiskit
- https://github.com/qiskit-community/ibm-quantum-challenge-africa-2021
- https://github.com/udvzol/option_pricing